Improving Text Classification using Knowledge in Labels

A Thesis Submitted to the Department of Computer Science and Communications Engineering,
the Graduate School of Fundamental Science and Engineering of Waseda University
in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Submission Date: January 25th, 2021

Department of Computer Science and Communications Engineering,
the Graduate School of Fundamental Science and Engineering,
Waseda University

Advisor: Prof. Hayato Yamana
Research guidance: Research on Parallel and Distributed Architecture

Student ID: 5119F058-3
Cheng Zhang

Abstract

Text classification, the most fundamental and vital task in natural language processing, has been widely studied and applied in many domains. Over the last few decades, surpassing results and progress have been achieved in this area due to the groundbreaking deep learning models. However, they rarely consider incorporating the additional knowledge hidden in the labels whether it is a traditional machine learning method or a deep learning approach. We argue that the additional hidden information in the labels leads to better text classification accuracy.

In this paper, instead of just encoding the labels into the numerical values, we incorporated the knowledge in the labels into the original model without changing the model architecture. We combined the output hidden states of an original classification model with the relatedness calculated based on the embeddings of a sequence and a keyword set. A keyword set is a word set to represent knowledge in the labels. Usually, it is generated from the classes while it could also be customized by the users. The experimental results show that our proposed method achieved a 1.7% maximum absolute increase among three popular text classification datasets compared with the original models like LSTM and BERT.

Contents

1 Introduction

Over the last few decades, text classification, one of the fundamental tasks in natural language processing (NLP), has been widely studied and applied in various domains in text mining, such as sentiment analysis [1][2][3], spam filtering [4][5], and topic labeling [6]. With the advent of the big data era, unstructured data, not structured in a pre-defined manner, is growing at an alarming rate per year and appears almost everywhere such as in social media, emails, and web pages. However, extracting insights from such kind of unstructured information, which is typically text heavy, is difficult and time consuming due to their convoluted nature. By adopting automatic text classification, unstructured text data can be analyzed quickly and cost-effectively compared with manual classification. Moreover, automatic text classification methods could yield more reliable and less subjective results which are useful for enhancing data-driven decisions.

A text classification dataset contains sequences of text as $D = \{S_1, S_2, \dots, S_N\}$, where each sequence is labeled with a class value. The label is usually shown as text while being converted to a numerical value for machine learning algorithms. For instance, classes in AG news dataset [7] {*world*, *sports*, *business*, *sci/tech*} will be transformed into {0, 1, 2, 3}. Although this translation is common in encoding labels, it neglects to incorporate hidden knowledge in the labels. The text classification model receives only these numbers without understanding their meanings. From a human perspective, number 1 represents the label "*sports*" related to football, competition, and entertainment while number 2 means the label "*business*" related to company, profit and money. Hence, taking the rich knowledge information hidden in the labels into consideration can lead to a better understanding between sequences and classes, thereby improving the performance of the text classification models.

Three main challenges exist in incorporating the knowledge in the labels into text classification models: (1) **Label Knowledge Selection**: Considering a given label, how to properly select its related information is difficult. For example, the label "*sports*" contains many forms of competitive physical activities, such as football, hockey, or racing. However, we cannot include all of them in the representation of the hidden knowledge. Too much related information may bring knowledge noise while insufficient knowledge incorporation leads few improvements; (2) **Label Knowledge Encoding**: After obtaining related information from the labels, how to represent it in the modern text classification models is an important problem; (3) **Heterogeneous Information Fusion**: Designing an appropriate method to fuse the original model's information and the knowledge information in the labels is another challenge.

To overcome the above challenges, we propose an effective approach to incorporate the knowledge in the labels into the model while without the need of changing the original model architecture. In our approach, label-related knowledge is represented by a keyword set that can be customized by the users. The main contributions of this paper can be summarized as follows:

1) This paper proposes a novel idea about utilizing the hidden knowledge in the labels.
2) The implemented new method significantly outperforms the original models while without changing the original model's architecture.
3) The codes of our proposed method are publicly available at https://github.com/HeroadZ/KiL.

In this thesis, studies related to text classification are introduced in Chapter 2. Details of the proposed method are presented in Chapter 3. We then described the experiments and results in Chapter 4. Finally, we provide concluding remarks and further optimization discussion in Chapter 5.

2 Related Work

Many efforts have been made to improve the performance of automatic text classification. Previous work on automatic text classification can be classified into two main systems: rule-based and machine learning-based.

Rule-based systems classify texts based on word patterns like counting the number of the specific words. Rule-based systems are comprehensible by humans and can be improved over time. However, designing rules for a complex text classification task can be quite challenging and time-consuming. Moreover, these handcrafted linguistic rules are difficult to maintain and do not scale well for texts from other domains.

On the contrary, machine learning-based systems classify texts based on the past observations. Machine learning algorithms can learn the potential associations between texts in different classes by using pre-labeled examples as training data. For the machine learning-based systems, there are mainly two types: traditional machine learning models and deep learning models. There are three main differences between them. First, traditional machine learning models have a rather simple structure while deep learning models are, like a human brain, complex and intertwined. Secondly, traditional machine learning models rely heavily on feature extraction, while deep learning models need only little human intervention. Thirdly, deep learning models require much more data and computation power than traditional machine learning models to function properly.

2.1 Traditional Machine Learning Models

Traditional machine learning models dominated in the field of text classification since the 1960s. This method significantly outperforms the rule-based system in accuracy, stability, and scalability. Feature extraction and classifier selection are two essential parts for a traditional machine learning model.

2.1.1 Feature Extraction

Because text data cannot be directly used for classification, feature extraction is necessary to transform text data into numerical data for the model. In this step, data cleaning functions like stop words removing are optional. The common techniques for feature extraction are listed as follows.

- *Bag-of-words (BOW)* [8]: representing a text by a vocabulary of known words without considering the order of words.

- *Term frequency (TF)* [9]: reflecting the importance of words by using a weighting matrix based on the number of occurrences of terms. It can be calculated by equation (1) where $f_{t,d}$ represents the raw count of term t in document d.

$$tf(t,d) = \log_e(1 + f_{t.d}) \tag{1}$$

- *Term frequency-inversed document frequency (TF-IDF)* [10]: reflecting the importance of words by multiplying term frequency (TF) and inverse document frequency (IDF). IDF is for preventing the high weights of common high-frequency words such as *"the"*, *"this"*, *"and"*. The formula of TF-IDF is shown in equation (2) where D represents all documents in the corpus.

$$\begin{aligned} tf\ idf(t,d,D) &= tf(t,d) \cdot idf(t,D) \\ &= \log_e(1 + f_{t.d}) \cdot \log_e\left(\frac{count(D)}{count(d \in D: t \in d)}\right) \end{aligned} \tag{2}$$

- *Word Embeddings* [11]: representing words in the vectors of real numbers by training on large corpora to quantify and categorize the semantic similarities between words. Famous previous work includes Word2Vec [11] and Global vectors for word representation (Glove) [12].

2.1.2 Classifier Selection

The second part, which is the most important step, is choosing a proper classifier. Many classic classification algorithms have been designed since the 1960s. Here we briefly introduce several most popular classification algorithms based on different theories.

- *Decision trees (DT) -based* [13]: DT learns the correlation between classes and attributes by constructing the tree structure which reflects the theory of divide-and-conquer.

- *Probabilistic classifiers*: This kind of classifiers primarily use the prior probability to calculate the posterior probability under the assumption that the features are independent like a Naïve Bayes classifier [14].

- *K-nearest neighbor (KNN) -based* [15]: The input will be classified by a plurality vote of k-nearest labeled neighbors.

- *Support vector machine (SVM) -based* [16][17]: SVM constructs an optimal hyperplane or set of optimal hyperplanes in a high-dimensional space by maximizing the distance between the hyperplane and the pair categories, consequently reaching the best ability of generalization.
- *Ensemble learning classifiers*: Ensemble algorithms, such as random forest (RF) [18], AdaBoost [19] and XGBoost [20], aims for obtaining better predictive performance and interpretation by aggregating the results of multiple algorithms.

2.2 Deep Learning Models

Deep learning models have achieved superior results than the traditional machine learning models and have become the mainstream method for the text classification task since the 2010s. Main deep learning architectures including:

- *Multilayer perceptron (MLP) -based* [21][22]: A MLP model, sometimes colloquially referred to as "vanilla" neural networks, consists at least three layers of nodes: an input layer, a hidden layer and an output layer. It can distinguish linearly inseparable data by utilizing backpropagation for training.
- *Convolutional neural networks (CNNs) -based* [23]: CNNs are originally proposed to deal with image classification problems. As the regularized versions of MLP, CNNs use a series of convolutional layers and pooling layers in the hidden layers to capture more complex patterns with lower complexity. Kim et al. proposed TextCNN [24] that achieves excellent results on multiple benchmarks with little hyperparameter tuning and static vectors.
- *Recurrent neural networks (RNNs) -based*: A RNN can capture the dependency of variable length sequences through recurrent computation. Long short-term memory (LSTM) [25], the improvement of RNN, are broadly used in speech recognition and machine translation because it effectively alleviates the gradient vanishing problem that can be encountered when training traditional RNNs.
- *Transformer-based* [26]: Pretrained language models (LMs) like bidirectional encoder representations from transformers (BERT) [27] achieve state-of-the-art performance on NLP tasks including text classification due to effectiveness on learning global contextual semantic representation. Such kinds of LMs, pretrained on a large corpus, can be directly used in the downstream NLP tasks by simple fine-tuning.

Since the appearance of BERT which is considered as the milestone in NLP, numerous techniques have been proposed to improve it. Recent work demonstrated that injecting additional knowledge information can enhance the original pretrained LMs. Zhang et al. [28] are the pioneers in this direction by fusing entity information in a knowledge graph (KG) with BERT. Liu et al. [29] proposed K-BERT which utilizes the knowledge of relations between entities and designs a soft position and visible matrix to limit the impact of knowledge noise.

However, their methods must modify the original model's inner architecture and neglect the knowledge hidden in the labels.

3 Proposed Method

Currently, to our best knowledge, systems that notice the importance of labels have not been found by far. In this research, we construct a novel framework to effectively incorporate the knowledge hidden in the labels for the text classification task. The knowledge is represented by the relatedness between tokens in a sequence and keywords generated from the labels.

3.1 Notations

We denote a text sequence as $s = \{w_1, w_2, \ldots, w_n\}$, where $w_i \in \mathbb{V}$ is an i-th word that appears in the sequence s, and n is the length of the sequence. The keyword set generated from the labels is denoted as $\mathbb{K} = \{k_1, k_2, \ldots, k_m\}$. Note that m has no relation with n. In this study, we adopted word-level tokens in English texts. The whole vocabulary, which implies all the tokens, is denoted as $\mathbb{V}$.

3.2 System Overview

The overview of our proposed framework is presented in Figure 1. As shown in Figure 1, it consists of four modules: the *embedding layer*, the *original model* (middle left), the *mean relatedness layer* (middle right), and the *linear layer*.

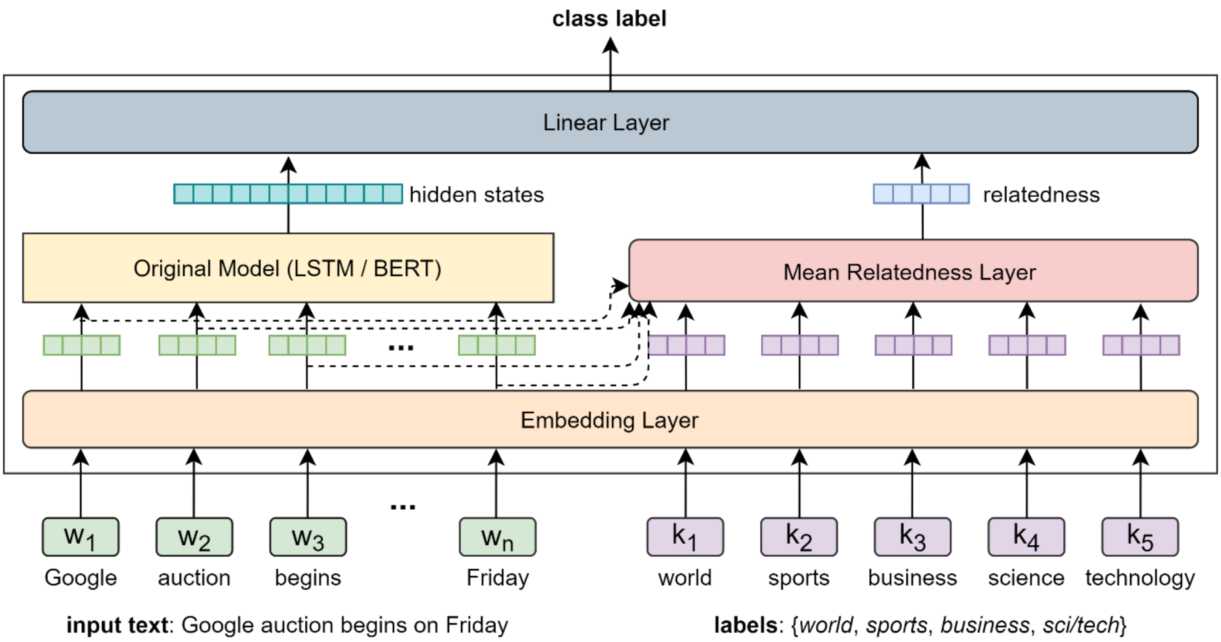

Figure 1. Overview of the proposed framework

- The *embedding layer*: a layer for transforming the numerical values into the vector representations.
- The *original model*: a text classification model such as LSTM or BERT.
- The *mean relatedness layer*: a layer for calculating the relatedness between tokens in a sequence and keywords generated from the labels.
- The *linear layer*: a layer for applying a linear transformation as shown in equation (3) to the input features. The output size of this layer is always equal to the number of the classes.

$$y = xA^T + b \tag{3}$$

3.3 Procedure

For a complete run, the following steps are conducted. Note that steps 3 and 4 can be performed parallelly.

1. **Tokenization**: The input sequence is transformed into the numerical values based on the vocabulary $\mathbb{V}$ which contains all tokens. As shown in Fig. 1, text "Google auction begins on Friday" could be tokenized into ["google", "auction", "begin", "on", "friday"] by performing lowercase conversion and lemmatization. Then these tokens might be transformed into [1, 2, 3, 4, 5] where the numbers represent the indices of the corresponding token in $\mathbb{V}$.

2. **Word Embeddings Conversion**: Instead of just feeding the indices into the model, the indices are mapped to corresponding word embeddings by the simple lookup table in the embedding layer. The reason is that the vocabulary is discrete, which means that there is no relationship behind the indices while the association between words is common. Therefore, a continuous, distributed vector representation of words can capture the semantic similarities between words, thereby improving the performance of models. For example, the word embeddings of tokenized indices [1, 2, 3, 4, 5] are noted as $\{e_1, e_2, e_3, e_4, e_5\}$ where $e_i \in \mathbb{R}^{1\times d}$ and d is the dimensionality of the word embeddings.

3. **Original Model Training**: Then we feed the word embeddings into a text classification model such as LSTM or BERT. The output of the model is hidden states in the hidden layer after the finish of training.

4. **Relatedness Calculation**: We calculate the relatedness for each keyword in $\mathbb{K}$. For the relatedness between one keyword and a sequence s, the mean value of similarities between this keyword and all tokens in s will be calculated. Simple dot product is performed for the calculation of similarities since the representation of words is vectors.

5. **Concatenation**: We simply concatenate the output hidden states from the original model after training and relatedness vectors since both are 1-dimension vectors.

6. **Classification**: The size of neurons should be shaped into the number of classes by applying a linear transformation. It would be better to apply softmax function in equation (4) to the output vectors of the last linear layer for text classification task. The class label can be obtained by retrieving the indices of the maximum value in the output vectors.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4}$$

3.4 Relatedness Calculation

The mean relatedness layer calculates the relatedness between a sequence s and each keyword in the keyword set $\mathbb{K}$. The detailed calculation process is shown in Fig. 2.

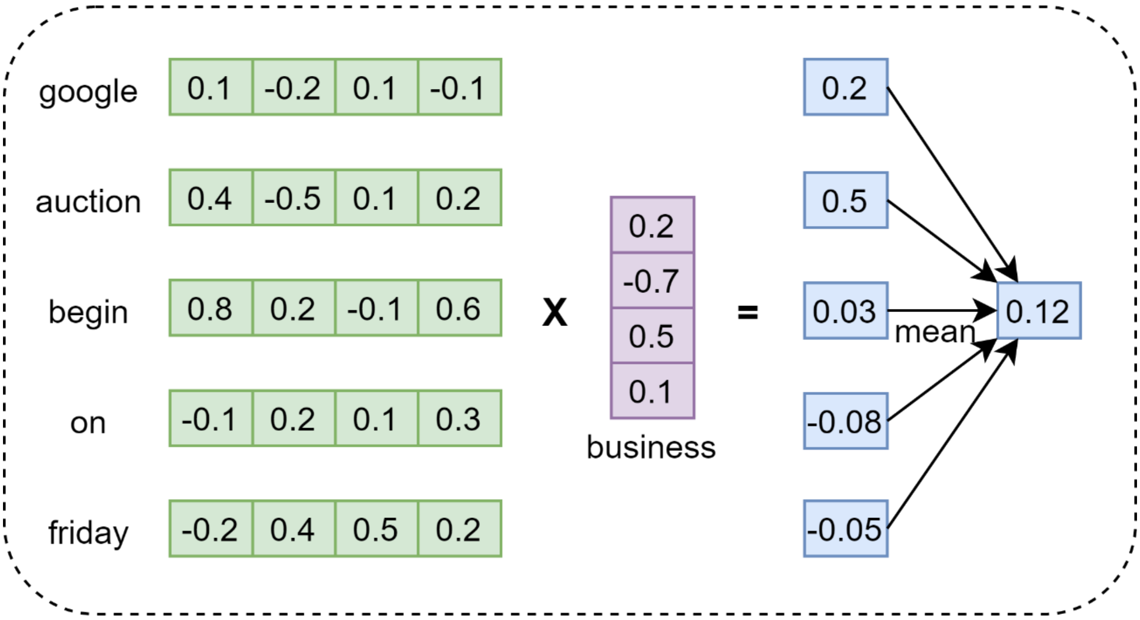

Figure 2. Calculation of relatedness between a sequence and a keyword.

Suppose that there is a sequence s and a keyword set $\mathbb{K}$ after the transformation by the embedding layer. Note that the word embeddings of s are represented as $\{e_{w1}, e_{w2}, \dots, e_{wn}\}$ and $\mathbb{K}$ is represented as $\{e_{k1}, e_{k2}, \dots, e_{kn}\}$, where $e_{wi}, e_{ki} \in \mathbb{R}^{1\times d}$ and d is the dimensionality of the word embeddings. The common dimensionality of pre-trained embeddings, such as Glove, is 300 and 768 for the base BERT. In Fig. 2, the dimensionality of the word embedding is set as 4 for a simpler understanding. Then, we calculated the similarity between each token and each keyword using a dot product as described in equation (5).

$$a \cdot b = \sum_{i=1}^{n} a_i b_i \tag{5}$$

In Fig. 2, the similarities between the keyword "*business*" and tokens {"google", "auction", "begin", "on", "friday"} were calculated. Then, the relatedness between the sequence s and the keyword "*business*" was obtained by the mean function. The relatedness of other keywords was calculated in the same way. Therefore, the relatedness matrix of each sequence s is $r \in \mathbb{R}^{1\times m}$, where m is the length of $\mathbb{K}$. For a better explanation, Algorithm 1 shows the pseudo code of the detailed implementation.

Algorithm 1 *Relatedness Calculation*

Input word embeddings of tokens $\boldsymbol{E_w} = \{\boldsymbol{e_{w1}}, \boldsymbol{e_{w2}}, \dots, \boldsymbol{e_{wn}}\}$,
word embeddings of keywords $\boldsymbol{E_k} = \{\boldsymbol{e_{k1}}, \boldsymbol{e_{k2}}, \dots, \boldsymbol{e_{kn}}\}$

Output relatedness vector $\boldsymbol{r} \in \mathbb{R}^{1\times m}$

for e_{ki} in E_k **do**
  for e_{wj} in E_w **do**
    $temp[j] \leftarrow e_{ki} \cdot e_{wj}$
  end
  $r[i] \leftarrow mean(temp)$
end
return r

4 Experiments

4.1 Datasets

In the experiments, we used three widely used text classification datasets in English to evaluate the effectiveness of our proposed approach: AG news[1], IMDB movie reviews[2], and 20 newsgroups[3]. An overview of these datasets is described in Table 1. The reasons for choosing these datasets are summarized below:

- They are all widely used by other researchers and easy to obtain.
- AG news and IMDB reviews differ in the size of data; however, they both have a small number of classes. Therefore, we could observe the effect of the dataset size on our proposed method.
- IMDB reviews and 20 newsgroups differ in the number of classes, while similar in data size. Consequently, the effect of the number of classes can be noticed.

Table 1. Overview of datasets used

Dataset	Train size	Test size	Number of classes
AG news	120k	76k	4
IMDB reviews	25k	25k	2
20 Newsgroup	11,314	7532	20

4.2 Baselines

In this research, we compared our proposed methods with two baselines:

- Two-layer bidirectional LSTM [25] with 300-dimensionality pre-trained embeddings

[1] http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html
[2] https://ai.stanford.edu/~amaas/data/sentiment/
[3] http://qwone.com/~jason/20Newsgroups/

Glove[4] [12], which was trained on Wikipedia 2014 and Gigawords 5th edition[5].

- 12-layer, 768-hidden, 12-heads, 110M parameters pre-trained base uncased BERT [27], which is provided by HuggingFace[6].

4.3 Metric

The metric we used in this paper is accuracy, which is given in equation (6). Accuracy is the most common criterion for measuring the performance of text classification systems.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{All\ Samples} \tag{6}$$

4.4 Parameter Settings

4.4.1 Keyword Settings for Datasets

For each dataset, we defined keywords that could represent the knowledge in the labels. Here, we only used keywords extracted from labels without additional words. For instance, we only used {positive, negative} as keywords for IMDB movie reviews, while words such as {good, awesome, boring} might also be helpful for better label representation.

- AG news (5): {world, sports, business, science, technology}
- IMDB movie reviews (2): {positive, negative}
- 20 newsgroups (31): {atheism, computer, graphics, microsoft, miscellaneous, system, hardware, windows, mac, ibm, sale, recreation, entertainment, auto, motorcycle, sports, baseball, hockey, science, cryptography, electronics, medical, astronomy, space, social, religion, christian, gun, politics, arab, mideast}

4.4.2 Parameter Settings for Base Models

For LSTM, we set the dropout ratio to 0.2. As the data sizes of different datasets vary, the learning rate and batch size should be adjusted for stable training, such as batch sizes of 2048 for AG news dataset and 32 for 20 newsgroups dataset. The training epoch for all

4 https://nlp.stanford.edu/projects/glove/
5 https://catalog.ldc.upenn.edu/LDC2011T07
6 https://huggingface.co/transformers/pretrained_models.html

datasets was set to 20.

For BERT, Devlin recommends the range of fine-tuning parameter as follows[7]:

- Batch size: 8, 16, 32, 64, 128
- Learning rate: 3e-4, 1e-4. 5e-5, 3e-5
- Epoch: 2, 3, 4

Thus, we used 5e-5 as the learning rate for all datasets. An epoch of 3 was set for AG news dataset, while an epoch of 4 was set for 20 newsgroups dataset for data insufficiency.

4.5 Evaluation Results and Discussion

The results are shown in Table 2. For LSTM, we took the mean accuracy of the top five epochs (left) and the highest accuracy (right) for comparison, as several epochs are necessary for training a stable model. On the contrary, BERT had a stable accuracy as fine-tuned BERT results were stable owing to its complexity. The proposed and original methods had the same parameter settings. The better results are in bold.

Table 2. Evaluation Results on Accuracy (%)

Dataset	**Method**	**Model**	
		LSTM (*)	***BERT***
AG	original	92.50 / 92.62	94.64
	proposed	**92.72 / 92.82**	**94.72**$^{+}$
IMDB	original	88.78 / 88.91	93.93
	proposed	**89.53 / 89.68**	**94.06**$^{+}$
20 Newsgroup	original	79.18 / 79.73	86.27
	proposed	**80.84**$^{++}$ **/ 81.35**$^{++}$	**87.24**$^{++}$

(*) Mean accuracy of the top five epochs / the accuracy of the highest epoch

+: statistically significant for $p < 0.05$.

++: statistically significant for $p < 0.01$

As shown in Table 2, we have the following observations:

[7] https://github.com/google-research/bert

- Our proposed method outperformed the original model, particularly for the smaller datasets with a larger number of classes, i.e., Newsgroup (maximum absolute accuracy increase of 1.7%).
- The improvement for BERT was smaller than that for LSTM. The reason might be that BERT has already had a better understanding for text owing to its capacity to handle complex parameters.
- More keywords resulted in higher improvements. We have 5 keywords for AG 5, 2 keywords for IMDB, and 31 keywords for Newsgroup.
- We could not confirm the statistical significance for AG and IMDB when adopting LSTM.

4.5.1 Relationship between improvements and training data size

The relationship between improvements and training data size has been investigated by taking 10%, 20%, …, 100% of the training data. Here 20 Newsgroup dataset was used as an example. For a more stable result, we randomly selected fixed percentage data from the training data for 10 times and took the average value of 10-time results. The randomly selected training data were used both in original and proposed models. The results are shown in Figure 3. We confirmed that the improvements on accuracy are independent from the training data size with the same keywords. The improvements on BERT are smaller than that on LSTM.

4.5.2 Relationship between improvements and number of keywords

We also investigated the relationship between improvements and the number of keywords by taking different number of keywords. Here we also used 20 Newsgroup dataset as an example. The total number of keywords of 20 Newsgroup dataset is 31. In the experiment, {5, 15, 25} was set as the number of keywords on the same training data. Similarly, we randomly selected fixed number of keywords from the keywords set for 10 times and averaged the 10 results for each number. The results are shown in Figure 4. There is a linear relationship between the number of keywords and the improvements in Figure 4, i.e., more keywords bring more improvements if the keywords are not noise.

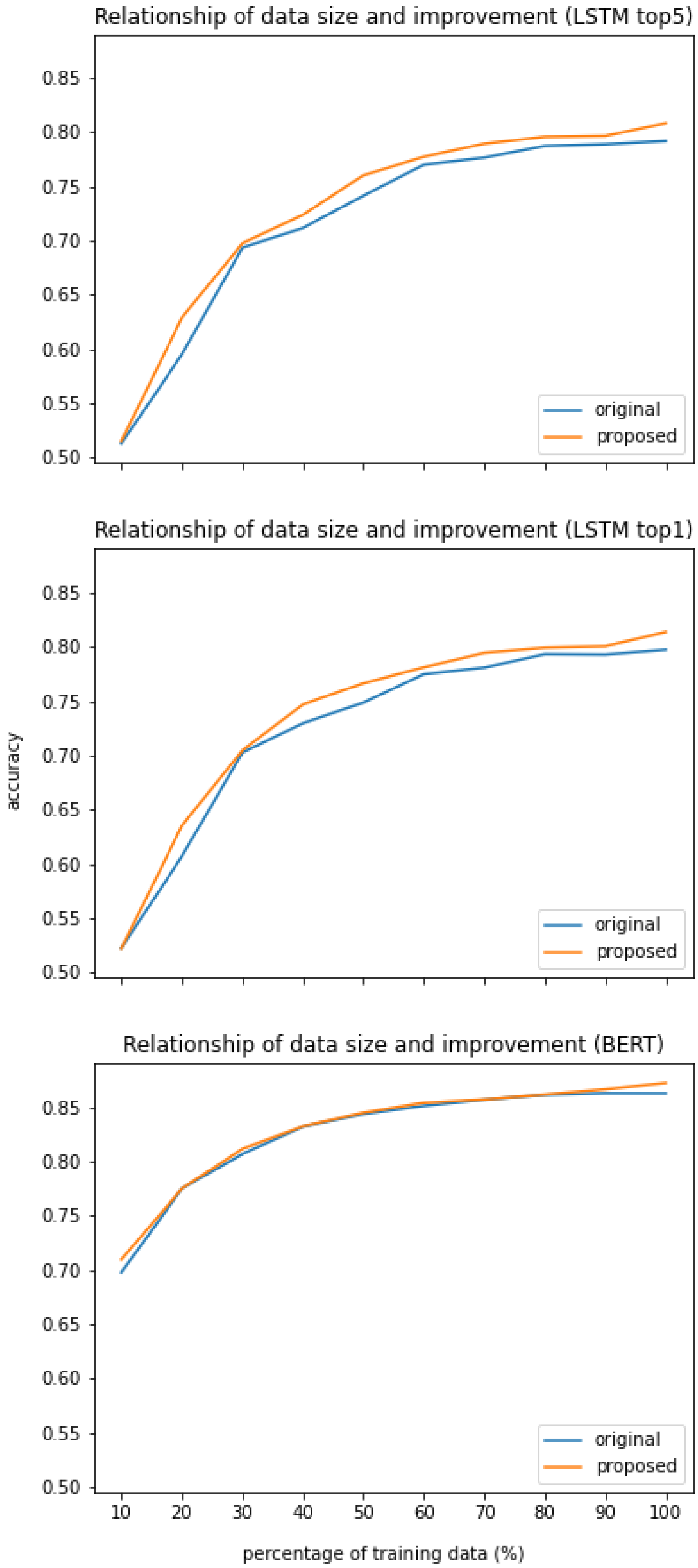

Figure 3. Relationship between improvements and data size.

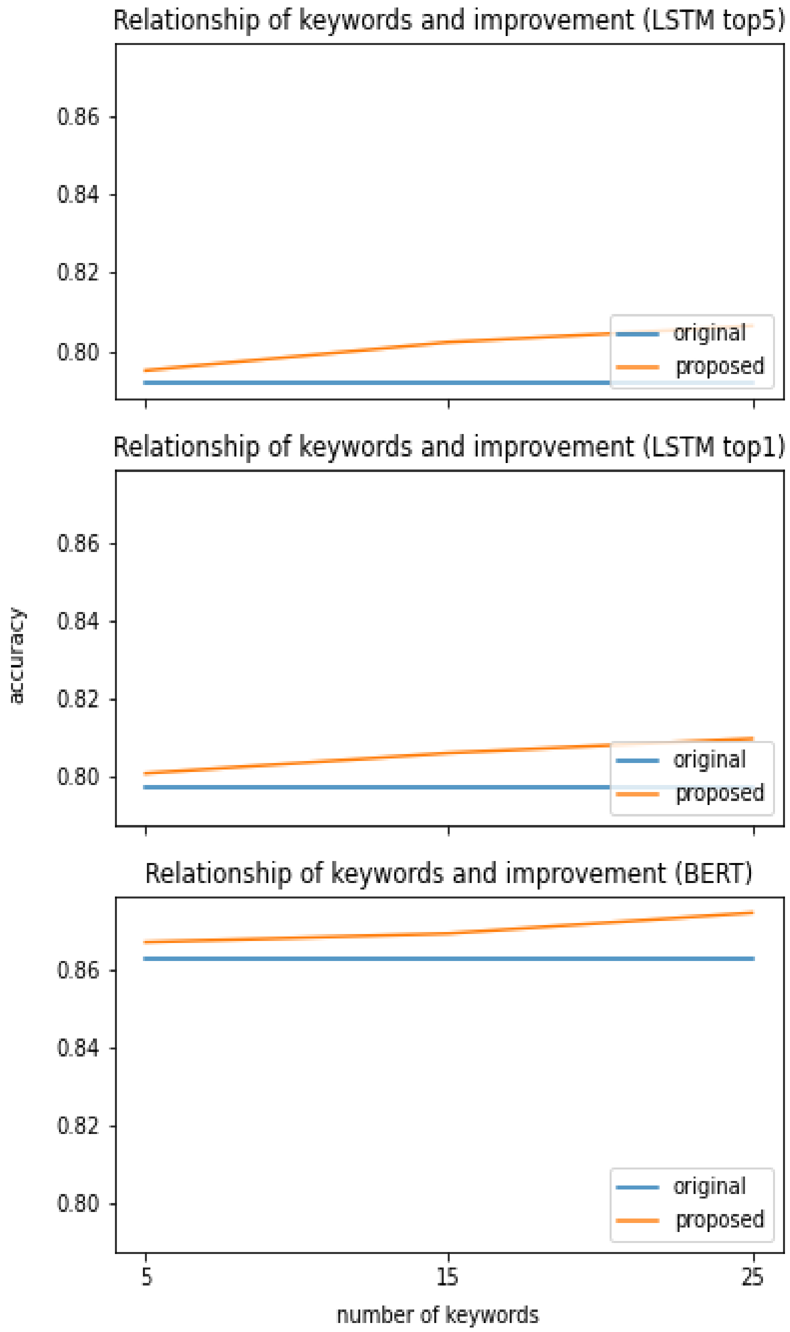

Figure 4. Relationship between improvements and number of keywords

5 Conclusion

This thesis proposed an effective approach to incorporate knowledge in the class labels into text classification models. We designed a keyword set that could be customized by users to represent the knowledge in the labels by calculating the relatedness between text sequences and keywords. The experimental results demonstrate that our proposed method is capable of understanding the relationship between sequences and labels. Furthermore, our proposed method performed well on datasets with many classes. An absolute accuracy increases of 1.7% was obtained for LSTM and 1.0% for BERT on the Newsgroup dataset.

Future work consists of two important directions: (1) creating a proper keyword set to represent knowledge in labels without adding knowledge noise and (2) finding a better calculation method for relatedness, not just using the mean value. These directions may lead to more general and effective label understanding and text classification.

Acknowledgement

First and foremost, I would like to express my deep and sincere gratitude to my supervisor, Prof. Yamana, for providing invaluable guidance and continuous support during my years in graduate school. His vision, open-mindedness, patience, and enthusiasm have deeply inspired me. I have gotten many suggestions on research like how to make a great survey or conducting a statistical test on the experimental results.

I am also grateful to Masashi Kudo, Takuya Suzuki, Fan Mo, Huida Jiao, Shun Morisawa for their self-giving help in my research. They taught me a lot like how to tackle problems occurred in the experiment or how to improve my academic writings, etc.

And finally, last but by no means least, also to everyone in the yamana lab. It was great to learn various knowledge from you and discuss with you.

Publication

1. Cheng Zhang, Masashi Kudo and Hayato Yamana. “Evaluation of BERT and XLNet Models on Irony Detection in English Tweets,” *DEIM2020*, G1-4, pp. 1-7 (2020.3)

2. Cheng Zhang and Hayato Yamana. “WUY at SemEval-2020 task 7: Combining BERT and Naive Bayes-SVM for humor assessment in edited news headlines,” In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, (2020.12)

3. Cheng Zhang and Hayato Yamana. “Improving Text Classfication using Knowledge in Labels,” in *Proceedings of 6th IEEE International Conference on Big Data Analytics (ICBDA 2021)*, (accepted)

References

[1] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. *In Proceedings of the49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[2] W. Medhat, A. Hassan, and H. Korashy. Sentiment analysis algorithms and applications: a survey. *Ain Shams engineering journal*, 5(4):1093–1113, 2014.

[3] R. Feldman. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89, 2013.

[4] T. S. Guzella and W. M. Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, 2009.

[5] L. Zhang, J. Zhu, and T. Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004.

[6] S. Hingmire, S. Chougule, G. K. Palshikar, and S. Chakraborti. Document classification by topic labeling. *In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 877–880, 2013.

[7] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. *In Advances in neural information processing systems*, pages 649–657, 2015.

[8] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[9] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523,1988.

[10] J. Ramos et al. Using tf-idf to determine word relevance in document queries. *In Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. New Jersey, USA, 2003.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Dis-tributed representations of words and phrases and their compositionality. *In Advances in neural information processing systems*, pages 3111–3119, 2013.

[12] J. Pennington, R. Socher, and C. D. Manning. Glove: global vectors forward representation. *In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[13] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[14] I. Rish et al. An empirical study of the naive bayes classifier. *In IJCAI2001 workshop*

on empirical methods in artificial intelligence, volume 3 of number 22, pages 41–46, 2001.

[15] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[16] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[17] T. Joachims. Text categorization with support vector machines: learning with many relevant features. *In European conference on machine learning*, pages 137–142. Springer, 1998.

[18] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[19] Y. Freund and R. E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[20] T. Chen and C. Guestrin. Xgboost: a scalable tree boosting system. *In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[21] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386,1958.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[24] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv*: 1408.5882, 2014.

[25] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N.Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics, June 2019.

[28] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. ERNIE: enhanced language

representation with informative entities. *In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics, July 2019.

[29] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang. K-bert: enabling language representation with knowledge graph. *In Proceedings of the 34th AAAI conference on artificial intelligence*, pages 2901–2908, 2020.